This is the second lecture explains what other information is stored in the stack memory.

Let us review the previous lecture. When a function is called, the line number after this call is stored in the stack memory. This is called the return location because it tells the program where to continue after the function finishes.

In many cases, a function is called with one or several arguments. What is an argument for a function? It is the input of the function. By providing different values to the argument (or arguments), the function may behave differently. This is good because the function can do different things based on the input arguments.

When a function is called with one argument (or several arguments), the value of the argument is store in the stack memory. In this example, function f2 has one argument y and the function call at line 4 assigns 7 to the argument. Thus y’s value is 7 inside function f2.

If a function has several arguments, they are stored in the stack memory. In this example, function f3 is called with two arguments: 7 and 3.2. As a result, y’s value is 7 and z’s value is 3.2. These two arguments are stored in the stack memory.

Let us review what we have learned so far:

When a function is called, the statement after the call is stored in the stack memory.

If the call has an argument, the argument is stored in the stack memory. If the function call has multiple arguments, all of them are stored in the stack memory.

Before I talk about more concept, I want to share with you an important idea writing better programs. You want to make each function’s behavior depends on the input argument (or arguments) only and nothing else. If this function is called multiple times with the same values for the arguments, the function should do exactly the same thing.

Thus, the function must not have any static variable. The problem of static variables is that they keep the values from the previous execution of this function.

The program must not have any global variable. The problem of global variables is that they may be changed by another function.

Static and global variables may make a function behave differently even though the input arguments are unchanged. This can be very confusing when you test your program.

The reason is that static variables and global variables make a program difficult to understand and difficult to debug. Please read this short, but influential paper, “Global Variables Considered Harmful” published in 1973. The basic idea is that you need to keep track of the changes of static and global variables. If you lose track, understanding a program becomes very difficult and mistakes are more likely to occur.

Now, let’s review what we have learned so far about the stack memory. For each function call, the stack memory stores the return location and the arguments’ values.

These two pieces of information is called the frame of the called function. As you can see in this example, a frame of f3 includes the information about the return location and the arguments’ values.

Next, we consider the called function’s local variable (or variables). The local variable is also stored in the stack memory.

The stack memory can have three things: The first is the return location. This is the line number after the function call. The second is argument (or arguments). This is the input to the function. The third is the local variable (or variables).